Understanding StackOverflow Posts using FLASH and SVM as DUO

CSC791 Foundation of Software Science 2018

Fahmid Morshed Fahid
North Carolina State University
Raleigh, NC, USA
fahmid.morshed.fahid@gmail.com

ABSTRACT

This decade has seen a significant contribution in Machine Learning. But, there is also an enormous demand of resource usage in most cases. In the Text Mining domain, Deep Learning, for example, demands days of CPU time while identifying the relationship between different StackOverflow posts. Few studies showed that, a simpler approach is possible by combining optimizer and data miner. Our work explores this region by using optimizer to tune the hyperparameters of data miners that demands less CPU time without sacrificing the results. We extended the work of Majumdar[7] and Fu[5] by implementing FLASH as a hyperparameter tuner for SVMs on both clustered and unclustered data. We have also tried to produce a new dataset for Text Mining. In this study, we have found that, FLASH is a simpler and cheaper optimizer than DE that generates similar results as complex Machine Learning algorithms in significantly less CPU time.

KEYWORDS

Text Mining, Hyperparameter Optimization, Data Miner, Clustering, Word Embedding

1 INTRODUCTION

In contemporary computer science, Data Mining and optimization are considered to be two different problem sets. They both have different set of tools and methodologies. Data miners are used for classification and regression based problem based on a single dependent variable. By calculating variance or entropy, data miners either try to label a data point or try to find commonality between them. A Decision Tree, for example, is used to classify a data-point based on different attributes and their entropy. optimizer, on the other hand, knows how to find a Pareto Frontier[1] based on different goals irrespective of the goal types. An optimizer searches through the space and tries to find the set of optimum solutions. Genetic Algorithm, for example, tries to generate new generations by mutation and crossover to find the best fit for an objective function.

Inherently, optimizers are resource hungry but have nothing to do with the commonality of patterns in the dataset. On the other hand, data miners, being fast and robust, understands a dataset, but do not understand goal properly. Recent studies show that, Data miners can use optimizers to enhance their capabilities. For example, an optimizer can decide the depth and breadth of a Decision Tree for a specific dataset[17]. Alternatively, optimizers can also use data miners to improve the optimization process, either by using low CPU or low memory. For example, FLASH, an optimizer, uses a popular Data miner called CART[11]. This particular practice is called data miners Using or Used by optimizers, DUO (Dr. Menzies, Keynote Speaker, FSE 2018).

Traditional data miners are not optimized for a dataset and needs to be tuned. A long time practice is to tune a data miner manually, running the data miner multiple times on the same dataset and changing the parameters to find an optimum solution. A more advance method is to use Hyperparameter optimization[2] where the optimizers try to tune magic parameters of data miners to adjust the data miner to a particular dataset. Xia et al. showed that, the default set of parameters are no good for a data miner and should be tuned[17]. According to Xia et al., a combination of one or more optimizers can increase the predication of a Classifier significantly. But, to find the best magic parameters is an expensive process. As the combination of parameters can make the search space large, using Grid-Search on all possible combination of parameters for a classifier is very expensive. A more robust answer to this problem is random search[2]. After studying 8 dataset, Bergstra et al. found that, Random Search can find at least as good model as Grid Search with less iterations.

With this hyparparameter tuning in mind, many problems, that was in the domain of Deep Learning[3] for over a decade, can now be solved using DUO approach, with significant improvement in resource usage. Understanding post similarities of StackOverflow, for example, was first done by Xu et al. using Convolutional Neural Network (CNN)[18]. Fu et al. showed that, we can generate almost the same results using DUO instead of CNN, but with significantly

[1] Pareto efficiency or Pareto optimality is a state of allocation of resources from which it is impossible to reallocate so as to make any one individual or preference criterion better off without making at least one individual or preference criterion worse off. The Pareto frontier is the set of all Pareto efficient allocations, conventionally shown graphically. It also is variously known as the Pareto front or Pareto set. - Wiki

[2] In machine learning, Hyperparameter optimization or tuning is the problem of choosing a set of optimal Hyperparameters for a learning algorithm. The same kind of machine learning model can require different constraints, weights or learning rates to generalize different data patterns. These measures are called Hyperparameters, and have to be tuned so that the model can optimally solve the machine learning problem. Hyperparameter optimization finds a tuple of Hyperparameters that yields an optimal model which minimizes a predefined loss function on given independent data.

[3] Deep learning (also known as deep structured learning or hierarchical learning) is part of a broader family of machine learning methods based on learning data representations, as opposed to task-specific algorithms.

Foundations of Software Science, Fall 2018, Computer Science

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$9999.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

less CPU time[5]. Fu et al. used SVMs to find post similarities using the same StackOverflow dataset as Xu et al., but tuned those SVMs using an optimizer before classifying. Although, Differential Evolution (DE) is costly optimizer, but in comparison to CNN, a combination of DE and SVM shows at least a factor of 100 improvement in runtime.

Continuing in this direction, Majumder et al., showed that, at least 500+ times faster runtime can be achieved by using the DUO approach of Fu et al. with simple traditional clustering. This is essentially the other part of the DUO approach, using data miner for better optimization. This also signifies that, not all problem needs deep learning and if possible, we should avoid costly methods with simpler ones. In his work, Majumder et al. used the same Deferential Evolution (DE) algorithm to optimize SVMs, but after clustering the dataset locally using KMeans. To find the optimum number of clusters, they have used GAP algorithm[7].

Searching through the search space is costly for an optimizer. Most of the optimizer tries to creat better generation of data by mutating and doing cross-over between the data. FLASH is an optimizer that can find a Pareto Frointire on a large search space and also, can optimize on multiple goals by building simple CARTs[10]. As this algorithm has a built-in data-minier to guide in a large search space, FLASH takes less number of evaluation to reach the optimum goal.

Our project tries to continue the work of Fu et al.[5] and Majumdar et al.[7] to find better solution towards classifying StackOverflow dataset. SVMs, having unbounded configurations for tuning, takes a long time for DE to tune. FLASH, on the other hand, is a simpler surrogate model that can find the optimum goal faster with less iteration. We propose to use FLASH to tune SVMs for predicting StackOverflow posts because we believe, FLASH will be more simpler and faster than Differential Evolution. In other words, our project focuses on StackOverflow post classification problem and tries to answer the following questions:

(1) **RQ1:** Can we produce the same results as Majumdar et al. and Fu et al. using FLASH?
(2) **RQ2:** Can we improve the runtime of classifying StackOverflow posts?
(3) **RQ3:** Can we produce a new dataset and generate the same results?

Sarro et al. mentioned a few baselines for software engineering[14] and Dr. Menzies improved upon them to find a total of 16 baselines. The previous work of Majumder et al. achieved a few baselines such as Cheap and Context Aware. Our project targets a few more of the baselines from the list, in the following way:

(1) **Simple**: FLASH is simpler than DE to understand
(2) **Cheap**: FLASH is significantly faster than DE and uses less CPU and memory.

The following sections are organized as follows. Section 2 discusses the motivation behind our work, earlier works and the key concepts that will be needed to understand our work. In Section 3, we have explained our experiment at length. We have discussed about the dataset, the creation of a new dataset and the our methodologies and verification process of our results. Section 4 discuss our results. Section 5 is a discussion over the baseline that we have tried to improve as well as some insight into possible future work. Section 6 ends with conclusion of our work.

Note that, all our work is available at GitHub public repository[4]

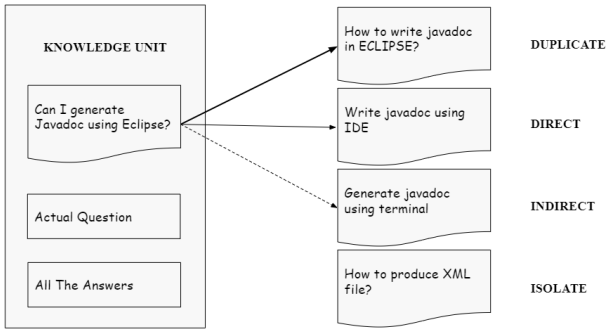


Figure 1: A visual representation of Knowladge Unit in StackOverflow and four types of relation between posts namely, (1)Duplicate, (2)Direct, (3)Indirect, (4)Isolate.

2 BACKGROUND

2.1 Problem and Motivation

StackOverflow is a privately held Question and Answers website for Coputer Science under Stack Exchange Network[5]. The website serves as a platform for users to ask and answer questions, and, through membership and active participation, to vote questions and answers up or down and edit questions and answers. This platform is particularly important for the developer community and vastly used to find different Computer Science related solution. Over the last decade, this website has turned into a public repository for software engineering problems. Many researcher believe that, a massive amount of knowledge can be harnessed by studying this website[1]

Finding related post in Stackoverflow is a tedious task. Often, the posts are duplicate, sometimes directly related and sometimes partially related. For someone in computer science, finding related post reduces the workload significantly. For example, if we find duplicate post on how to generate javadoc, we may suggest that for better understanding of the problem than reading a single post. Similarly, finding how to produce a xml file might lead someone to find about saving a pdf format and enhance their understanding about file handling in general. If we observe correctly, we can find four types of relations between two posts in StackOverflow, such as:

(1) Duplicate
(2) Direct
(3) Indirect
(4) Isolate

[4]https://github.com/FahmidMorshed/FLASH-SVM-StackOverflow
[5]Stack Exchange is a network of question-and-answer (Q&A) websites on topics in diverse fields, each site covering a specific topic, where questions, answers, and users are subject to a reputation award process. The reputation system allows the sites to be self-moderating. - Wiki

A single post can be linked to other posts by using one of the above mentioned type. Figure 1 shows an example of such linking types[18].

We believe, an automated system for finding link types between different StackOverflow posts will largely help the software development community for finding relevant knowledge and help developers to explore StackOverflow better. We also believe that, having a automated faster classifier for text mining in general will open other domains in text classifications.

2.2 Previous Work

Recent years in software analytic has a tendency to use more CPU to solve a simpler problem. Having cheaper hardware day by day, researcher are trying to use CPU extensive algorithm to generate results. But this kind of approach has been debated by researchers like Fisher et al.[4] over the years. Using more CPU does not always produce the best results and throwing CPU at a problem is not the only solution. We find that the slower the data mining method, the worse the user experience and the fewer the people willing to explore that method. These results are particularly acute in research where data miners have to be run many times to (a) explore the range of possible behaviors resulting from these methods or (b) generate the statistically significant results that can satisfy peer review.

According to our knowledge, Xu et al. were the first who tried to find the relationship between different StackOverflow Java posts. After building word2vec models using Gensim[12] n-gram model, they tried using Convolutional Neural Network. Although, they found an average F1 score of 0.85, their method took 50400 seconds on a single run to train the dataset[18].

But, consider the work Xu et al. on a standard validation loop as follows:

```
1. for 10 times in the project
2.     randomly divide the dataset into X=10 bins
3.     for each bin I in X
4.         train a CNN using (X–I)
5.         test I
```

Assuming line 4 takes about 50400 seconds to run, it takes about $10 \times 10 \times 50400 = 5040000$ seconds (about 58 days) for reproducing the work of Xu et al.

To solve such extensive CPU use, Fu et al. proposed a DUO approach towards the same problem. Instead of using a Neural Network, they have used Differential Evaluation (DE) algorithm to tune SVMs for the dataset. DE is an optimizing algorithm whereas SVMs are plain classifiers. But a SVM has four parameters namely C, karnel, gamma, coef0 and they all can be tuned. With better tuning, one can find a better classifier. Fu et al. tuned the parameters of SVM using DE. Fu et al. produced the same F Score as Xu did, but with significantly less time (about 65 times faster)[5].

Majumdar et al. enhanced this method by another step by introducing local learning. Instead of using a global SVM on the total dataset like Fu et al., they tried to learn multiple SVMs, each for a different cluster. As DE is an expensive algorithm, clustering it into smaller local group increases the performance of learning significantly. Majumar el al. first learned about the optimum cluster numbers using an algorithm called GAP[16]. According to Tibshirani et al., GAP can find the optimum number of clusters for KMeans clustering. After finding the number of clusters needed, Majumdar et al. clustered the dataset using Kmeans and for each cluster, they learned different SVM using DE. This local learning lost about 2% in F score over Xu et al. But, this approach significantly reduced the learning time and on single run on single core, it is about 500+ times faster[7].

Bayesian optimizer is one of the most popular optimizers. For multi-objective optimization, this optimizer tries to find the Pareto Frontier using Gaussian Process Model (GPM). A GPM outputs the mean and variation of a search-space and tries to predict a given function. Many multi-objective problems can be solved by using this Bayesian optimizer. A summary of all the possible variations of this optimizer has been introduced in the SigOpt'âĂŹs paper by Dewancker et al[3].

Bayesian Optimization has been used in many problems. One successful implementation of this optimizer can be found in the work of Hsu et al. where they tried to find best configuration for Cloud-based Virtual Machines. According to their study, Cloud VMs are not tuned accordingly and the default configurations for Cloud VMs are not efficient. For example, the could configuration for CPU utilization, working memory size or IO wait time should be tuned before using to find the maximum efficiency. They have also shown that different versions of Bayesian optimizer can be used to tune them and the search cost can be reduced[6].

Nair et al. showed a simpler approach for search based SE by introducing a algorithm called FLASH. This algorithm is a enhanced version of Bayesian optimizer but it can work for multiple objective. The core concept of their work was based on introducing an optimizer using a data miner. Instead of using Gaussian Process Model (GPM)[6], they have used CARTs to predict the uncertain part of the dataset and introduced a acquisition function called Bazza that does random vector projection to find the optimum goal on a multi-goal problem. This particular surrogate algorithm works significantly faster than state of the art algorithms like Bayesian optimizer and also can optimize for multi-objectives.[10]

Further, Nair et al. tried to implement FLASH on faster configuration optimization and found that, FLASH can optimize for single object optimization with large number of parameters. As flash only takes a small number of initial population and tries to predict the most uncertain (or certain) region in the search space, it takes less evolution to reach Pareto Frontier using FLASH. Nair et al. showed that, as the performance measure function is expensive, it is very costly for an optimizer to find the Pareto Frontier. But as FLASH needs only 60% of the evolution than a regular Residual Based Method to find the Pareto Frontier, it can easily find the optimum configuration very quickly. Also, FLASH can easily work on more than 10 configuration space whereas ePAL does not terminate[11].

Our project aimed to reproduce the work of Fu et al. and Majumdar et al. and to implement FLASH as a better optimizer and improve

[6]Gaussian process is a stochastic process such that every finite collection of those random variables has a multivariate normal distribution. The distribution of a Gaussian process is the joint distribution of all those random variables, and as such, it is a distribution over functions with a continuous domain. - Wiki

Parameter	Default	Tuning Options	Descriptions
C	1.0	[1, 50]	Penalty parameter of error term
Kernel	rbf	liner, poly, rbf, sigmoid	Specify the kernel type to be used in the algorithms Kernel
Gamma	$1/nFeatures$	[0, 1]	Kernel coecient for rbf, poly and sigmoid
Coeff0	0	[0, 1]	Independent term in kernel function. Only used in poly and sigmoid

Table 1: Different Parameters for SVM

the runtime creating a cheaper, simpler but robust algorithm. We tried to implement multi-core processing over the previous data to make the process run faster, if given the resources. Further, we tried to generate a new dataset to check our approach and verify FLASH as a better optimizer than DE for SVMs in Text Mining.

2.3 Key Concepts

To understand our project, here are some key concepts to follow. For better understanding, we encourage the reader to read through this sub-secion.

2.3.1 ***Word Embedding.*** Word Embedding is the method of converting words to vectors in order to compare their cosine similarities. Word embedding converts a word to a N dimension numeric vector. One popular word embedding is called continuous skip-gram model (word2vec). In this particular model, a two layer neural network is used to do unsupervised clustering that converts words into semantic vector representations [9]. This method is used by Fu et al. [5] and Xu et al. [18] in their research.

Word2vec learns the vector representation of a word (center word) by predicting surrounding words in a context window(c) by maximizing the mean of log probability of the surrounding words (w_{i+j}), given the center word (w_i) -

$$\frac{1}{n}\sum_{i=1}^{n}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{i+j}|w_i)$$

The probability $p(w_{i+j}|w_i)$ is a conditional probability defined by a softmax function[7] -

$$p(w_{i+j}|w_i) = \frac{\exp(v'^{T}_{w_{i+j}} v_{w_i})}{\sum_{w=1}^{|W|}\exp(v'^{T}_{w_i} v_{w_i})}$$

Here the v_w and v'^{T}_{w} are represents the input and output vectors of a word w in the neural network. W is the vocabulary of all words in the word corpus. $p(w_{i+j}|w_i)$ is normalized probability of word w_{i+j} appearing in a specific context for a center word w_i. Mikolove et al.[9] proposed hierarchical softmax and negative sampling techniques to improve computational efficiency, which can also be used for creating word embedding models.

In RQ1 and RQ2, our work uses the word2vec models trained by Fu et al. [5] that converted the StackOverflow text data into the corresponding vectors. Fu et al. used 100, 000 randomly selected knowledge units tagged with "Java" from Stack Overflow posts table (A knowledge unit is the combination of title, question and its corresponding answers). All the dataset has been preprocessed by them. We have used their preprocessed data to train and test our DUO approach.

For RQ3, we use 100, 000 randomly selected knowledge units tagged with "Python" from Stack Overflow posts table (include titles, questions and answers). This data was pruned by removing superflous HTML tags (while keeping short code snippets in code tag). We selected about half of the data to build the vocabularies of our word2vec models. Then fitted into the gensim word2vec module. This is a python wrapper over original word2vec package where, for word w_i in the post is sent to the trained word2vec model to get the corresponding word vector representation v_i . After converting, all the KUs the output vectors are then used for training and testing the models.

2.3.2 ***KMeans Local Clusters.*** While training the SVMs, one approach is to use the total dataset to train SVMs. In this approach, we split the data into ten bins, train our model on nine and test one the other one. This is essentially called a 10-fold cross validation. Another approach is to cluster the dataset into different bins and learn the model based on each clusters. This is called unsupervised learning. In this way, the training time reduces significantly. First, we find the similar data-point and cluster them. Then we train our model based on each cluster. Whenever a new data-point arrives for test, we first find out the cluster it belongs to and then use the model for that particular cluster. This approach performs almost least as good as global clustering (only 2% less). For example, Menzies et al. [8] shows that for defect prediction and effort estimation, models build on local part of data set at least as good as model build on global dataset, if not better.

One of the popular way to find the optimum number of clusters in a dataset for unsupervised learning is to use GAP statistic. Essentially, GAP tries to find the elbow in error while changing the number of clusters to find the optimum number. Majumdar et al. used GAP to find the number of clusters before doing KMeans to tune it better for the dataset. For more information on gap, see the work of Tibshirani et al.[16].

2.3.3 ***SVM.*** Support Vector Machine (SVM) is a type of Supervised Machine Learning algorithm that classifies the dataset by either using classification or regression. SVM use a hyperplane to divide the dataset into different section to classify them. SVM basically transfer the data into higher dimention so that the data point can be split into different parts and finds support vectors that essentially

[7] softmax function, or normalized exponential function, is a generalization of the logistic function that "squashes" a K-dimensional vector of arbitrary real values to a K-dimensional vector of real values, where each entry is in the range (0, 1) and all the entries add up to 1.

Algorithm 1 Differential Evolution

```
Require: n=10, cf=0.3, f=0.7
  frontier = random.guess(n)
  best = random.pick(frontier)
  lives = 1
  while lives > 0 do
    temp = state
    for i in |frontier| do
      old = frontier_i
      x, y, z = random.pick(frontier, 3)
      new = copy(old)
      for j in |new| do
        if random() < cf then
          new_j = x_j + f(z_j − y_j)
        end if
      end for
      if better(new,old)==False then
        new = old
      end if
      temp_i = new
      if better(new, best) then
        best = new
        lives + +
      end if
    end for
    frontier = temp
  end while
  return(best)
```

declears the boundaries between different classes. Emperical data show that SVMs are particularly suitable for Text Mining. In most cases, text mining data, converted to vectors are easy to differentiate using hyperplane and support-vectors.

SVMs have four different parameters. See Table 1 to understand all the different tuning that can be done on SVMs.

2.3.4 ***Tuning with Differential Evolution.*** Differential Evolution is a optimization algorithm that searches through a space to find the Pareto Frontier. It first tries to initialize a frontier of solutions. Then it tires to calculate an objective function. After that, using the initial frontier, the algorithm tries generate new set of candidates. To generate the new candidates for a better frontier, DE tries to extrapolate a solution based on the value f. Such extrapolations are performed for all attributes at probability cf. The candidates that are better than the ones in the current frontier get replaced. The search then repeats for the remaining frontier items. See the Algorithm 1 for better reference.

Fu et al. used DE to tune SVMs parameters showed in Table 1 to optimize for better F score. We tried to reproduce the same for our project. For the definition of "better", this study uses the same performance measures as Fu et al.; i.e. "better" means maximizing the objective score of the model based F1 Score.

2.3.5 ***Tuning with FLASH.*** FLASH is an optimizer that uses a CART data miner to guide for better goals. Essentially, this is a semi-supervised model that tries to predict better generation by observing a few using a CART. Nair et al. first introduced this algorithm[10]. FLASH tries to measure the goal of a handful of datapoints using the parameter called size. Then, untill budget runs out, it builds CART for each objective (in our case, only F score) and tried to predict the next best data point using the CART model. On each loop, it updates itself or looses life if it cannot. See Algorithm 2 for more details.

Algorithm 2 FLASH

```
Require: unevalConfig, fitness, size, budget
  evalConfig = [measure(x) for x in sample(unevalConfigs, size)]

  best = random(evalConfig)
  lives = 1
  while budget > 0 do
    model = CART(evalConfigs)
    acquired = measure(acquisitionFn(unevalConfigs, model))
    evalConfigs+ = acquired
    newBest = best(evalConfig)
    if newBest < best then
      budget − −
    else
      best = newBest
    end if
  end while
  return(best)
```

In our project, we tried to tune parameters of SVM using FLASH, for both clustered dataset and global dataset. We wanted to know if FLASH can produce the same results as DE but faster. Also, as FLASH is simpler than DE, we wanted to conclude that the whole approach to this particular problem can be simpler than the works of Majumdar et al. and Fu et al.

Majumdar et al. [7] used a local clustering model to improve the runtime of Fu et al. [5]

3 EXPERIMENT

Let us first define Knowledge Units. A Knowledge Unit is the collection of Question Title, description of the Question and all the related Answers in a StackOverflow post. See Figure 1 for a visual representation of a Knowledge Unit. This was defied by Xu et al. on their research in 2016[18]. We used the same defination throughout our project.

3.1 Dataset

3.1.1 ***Research Question 1 & 2.*** For Research Question 1 and 2 of our project, we have used the same dataset as Xu et al. did in their work. This data set contains all Java tagged posts. We will call this dataset Java dataset. This dataset contains 6,400 training examples of Knowledge Unit and 1,600 testing examples. For each type of links, we had 400 examples in our test set and 1,600 examples in our training set. For this reason, we did not use any data balancing technique. In the dataset Xu et al., they had marked two posts with a link type. There can be four link types in our dataset, namely Duplicate, Direct, Indirect and Isolate. So, our dataset, after

Id	PostId	RelatedPostId	LinkType	PostVec	RelatedPostVec	Output
0	205	130	1	VEC[...]	VEC[...]	VEC[...]
1	649	230	2	VEC[...]	VEC[...]	VEC[...]

Table 2: Dataset in Pandas Dataframe

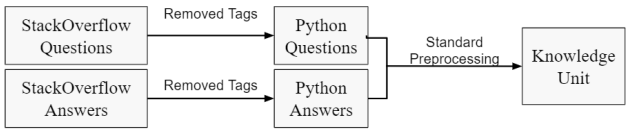


Figure 2: Building Knowledge Units.

converting them into Pandas Dataframe, had a post Id, related post id, and a link type.

For word embedding, Xu et al. also created ten different word2vec model using the vocabulary from the StackOverflow Knowledge Units. This word2vec models were used to convert the Knowledge Units into vectors. Note that, word2vec model converts words into vectors. We followed the procedure of Fu et al. and Majumdar et al. to create vectors of Knowledge Units. First, we preprocessed the data by doing tokenization, removing stop-words and stemming. Then each token was converted into vectors using word2vec model. Then we take the sum of all the tokens in a Knowledge Unit and took the mean value by dividing them with the word counts. This vector, essentially, represents one single Knowledge Unit. After finding the vectors of a post and related post, we took the average to get our final vector in our Output column. This vector is the relational value between two posts. A visual representation of our dataset after using word2vec models can be found in Table 2.

3.1.2 ***Research Question 3.*** For Research Question 3, we tried to build our own dataset. For that, we have collected all the StackOverflow posts tagged with 'Python' by using the data in Stack Exchange Archive[8]. We have also collected all those corresponding Questions and Answers from the same source. Then, we have combined the Questions and Answers. Then we have used BeautifulSoup[13] to remove the html tags and other unnecessary data. After that, we did standard Tokenization, removed Stop Words and did Stemming using Gensim Preprocessor. This is called standard Text Prepossessing. Finally we found our Knowledge Units (see Figure 2.

To build our test and training dataset, we have collected the "PostLinks.xml" from Stack Exchange Archive. This file basically includes two types of relations. Such as:

- Duplicate
- Direct

Note that, the link type 3 in this file is for Duplicate whereas we have used like type 1 as Duplicate. Similarly, like type 1 in this file is for Direct, whereas we have used link type 2 as Direct. This changes had to be made to follow our experiment with the previous dataset from Xu et al.

[8]https://archive.org/download/stackexchange

Algorithm 3 Mark Posts with Relation

Require: postIDs, relations
  $x, y = random(postIDs)$
  if (x,y) is NOT in relations **then**
    if y is in BFS(x) **then**
      $relations.insert(x, y, Indirect)$
    else
      $relations.insert(x, y, Isolate)$
    end if
  end if
  $return(relations)$

Then we took two posts at random from the dataset and checked if they were marked as Direct or Duplicate. If so, we leave them as it is. But if not, we did a Breath First Search to see if these two posts are indirectly related or not and thus mark them accordingly. See Algorithm 3 for more details. Note that, we marked indirect if two posts had at least a distance of two hops. It was an engineering decision from Xu et al.

Next we took random Knowledge Units to build a vocabulary for our word2vec models using Gensim library. This library is a cobination of two layer Convolutional Neural Network that uses skip-gram model. To avoid any kind of bias while building the vocabularies, we took only those knowledge units that were not in the Training or Test dataset that we have generated. As suggested by Xu et al., we build 10 word2vec models having feature length of 200 with random Knowledge Units.

Finally, we had the following two datasets.

(1) **Java Dataset:** Dataset from Xu et al. that were used in previous studies
(2) **Python Dataset:** Our own dataset produced from the Stack Exchange's StackOverflow Archieve.

3.2 Methodologies

Hyperparameter Tuning is Software Engineering domain has been neglected for a long time, due to the optimization run-time. But recent studies show that, the default parameters of data miners are no good and must be tuned for a better Classifier[17]. Fu et al. showed that, a standard DE can optimize SVM for predicting StackOverflow posts faster than a Neural Network. Then Majumdar et al. implemented local clustering to improve the runtime even further. In our project, we have implemented Fu et al. and Majumdar et al.'s work as baseline and checked their results with ours.

We first divided our data set into Test and Training set. Then, we converted them using a random word2vec model. After that, our work took three different steps. To produce the work of Fu et al. we used our training dataset in a 10-fold cross validation loop. On each time, we tried to tune SVMs using DE and predict the F score

		Predicted			
		C_1	C_2	C_4	C_4
Actual	C_1	$\underline{C_{11}}$	C_{12}	C_{13}	C_{14}
	C_2	C_{21}	$\underline{C_{22}}$	C_{23}	C_{24}
	C_3	C_{31}	C_{32}	$\underline{C_{33}}$	C_{34}
	C_4	C_{41}	C_{42}	C_{43}	$\underline{C_{44}}$

Table 3: Confuison Matrix

using our Test set. We did our experiment 10 times and generated 100 F scores and measured our run-time for each. We checked our results with the results of Fu et al. to verify our process. To produce the same results as Fu et al. these were the values for DE:

- Initial Population = 10
- cr = 0.3
- f = 0.7
- lives = 60

Note that, these values were suggested by Storn[15].

For generating Majumdar's result, after converting our Knowledge Units into vectors, we used our Training data on GAP algorithm to find the optimum number of clusters. Then, using this optimum number of clusters, we have used KMeans clustering to generated clusters. After that, for each cluster, we trained an SVM using a DE and each time, we predicted the Test set for F score. The test set essentially finds the closest cluster and then uses the corresponding SVM for that cluster to do the prediction. The parameters of DE as as before. The whole process was done using a 10 fold cross validation. We did 10 runs of this experiment to have better understandings of our results and matched them with Majumdar's work.

For implementing FLASH in both case, we have removed the DE. Then, we had to generate a search-space. We took random values for four parameters of SVM (C, Kernel, Gamma, Coef0) to build our search-space. We tried different sizes of search-space and finally, decided to use 500,000 data-points for tuning SVMs. Note that, generating more points do not improve our results, but also do not change our performance that is noticeable. But to ensure less memory, we tried to keep it as low as possible without loosing on our final F Score. We have implemented multicore support for our program using Multiprocessing library of python. Then we run our experiment with FLASH for both clustered dataset using KMeans and GAP as well as on unclustered dataset. For both cases, we have measured the F score and run-time using 10-Fold Cross Validation. We tried tuning FLASH with different values and the final values that we end up with are the following:

- Budget = 5
- Initial Population = 5
- Search Space = 500,000

3.3 Performance Measurement

For evaluiting our results, we have used F Score like Xu et al. did. We also measured Precision and Recall. But as previous studies had F Scores as their performance criteria, to compare with their results, we have chosen F Score as our measurement. Note that, it is a multi class classification problem (Duplicate, Direct, Indirect, Isolate). Thus, we used Macro F Score like our previous studies. This Macro F Score can be calculated by the confusion matrix[9]. The underlined value of the Table 3 are the True Positive. Using the cofusion matrix, we can calculate the F Score for individual class using the following equation:

$$precision = \frac{C_{ii}}{\sum_i C_{ij}} \tag{1}$$

$$recall = \frac{C_{ii}}{\sum_i C_{ji}} \tag{2}$$

$$FScore = \frac{2 \times recall \times precision}{recall + precision} \tag{3}$$

Now to calculate Macro F Score for all class, we calculate the F score of individual class and then divide them by the number of classes. We followed this, because this was used in all the previous studies.

3.4 Static Analysis

To validate our results, we have calulated the median values for the F scores and for our runtime. To test our results hold true, we have used two different static analysis on our results. We have used Effect Size test to confirm that, in F score, our result has no large variation from the past results. For runtime, this showed that, our result has large effect than the previous results. Then we have used Significant Test to verify that our F score is similar to the previous results and our runtime is significantly different than previous results. Note that, most of the distribution in real life do not follow any standard distribution, for example, Gaussian Distribution. To comply with that assumption, we have used non-parametric Cliff Delta as our Effect Size test and non-parametric Bootstrap as our Significance Test. Both of these algorithm do not assume any standard distribution of the data and thus the validation of our results should be true.

4 RESULTS

Let us discuss our results according to our Research Question presented earlier.

RQ1: Can we produce the same results as Majumdar et al. and Fu et al. using FLASH?

For RQ1, we have used the dataset from Xu et al. that has Java Tagged StackOverflow posts. We have used the same Differential Evolution as Fu et al. and Majumdar et al. did in their studies with the same initial parameters. In our own approach of using FLASH, we have tuned the parameters to produce the same results with minimum Run time. As we can see from the F Score column of Table 4, our result holds with earlier results in general. For better understanding our result, see Figure 5. To confirm our claim, we have also used static analysis. According to our results found, we can produce the same F Score as Fu did (93%) and for clustered data, we can produce the same F Score as Majumdar did (91%) using FLASH. Notice that, Majumdar has about 2% less F score than Fu, but gains a significant Run-Time improvement and in their work, they

[9]Confusion Matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in a predicted class while each column represents the instances in an actual class (or vice versa).

	Fu's DE SVM			Our FLASH SVM			Majumdar's KMeans DE SVM			Our KMeans FLASH SVM		
Class	Precision	Recall	**F Score**	Precision	Recall	**F Score**	Precision	Recall	**F Score**	Precision	Recall	**F Score**
Duplicate	91	92	**92**	90	93	**91**	91	90	**90**	92	88	**90**
Direct	91	90	**90**	93	91	**91**	90	88	**89**	87	88	**88**
Indirect	90	98	**95**	97	97	**97**	96	97	**97**	96	98	**96**
Isolate	91	93	**92**	95	90	**94**	90	91	**91**	91	92	**91**
Average	91	93	**93**	93	94	**93**	92	91	**92**	92	91	**90**

Table 4: Java Dataset Result Comparison (RQ 1 and 2)

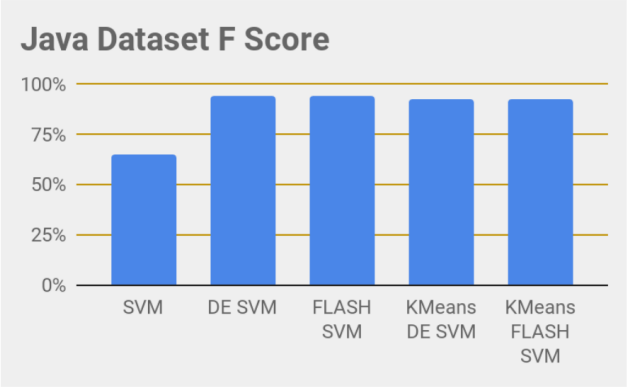


Figure 3: Java Dataset F Score Comparison (RQ 1 and 2)

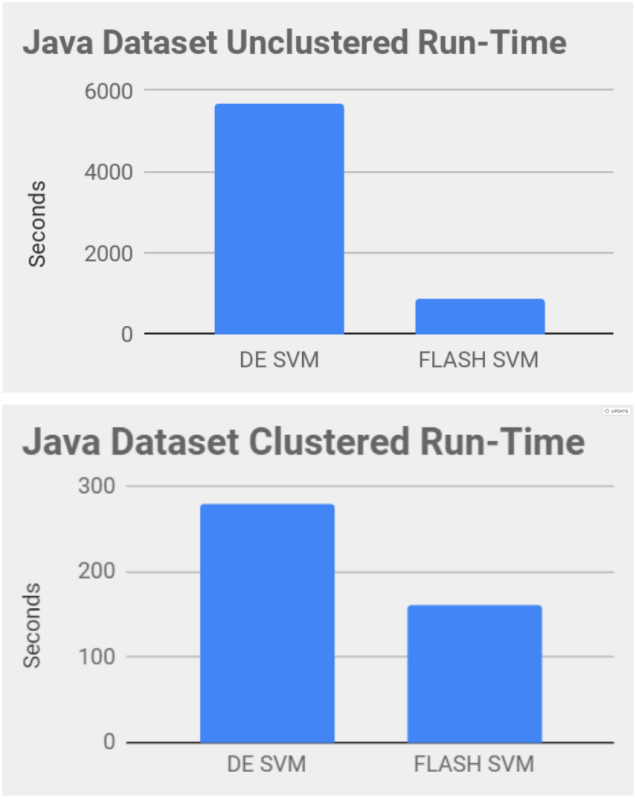


Figure 4: Java Dataset Runtime Comparision (RQ 1 and 2). Unclustered DE SVM 5680s, Unclustered FLASH SVM 845s, Clustered DE SVM 280s, Clustered FLASH SVM 160s. To understand the relative improvement, two different charts were used.

have argued the trade-off at great detail. So, to answer our Reseach Question 1, we can say that, we can produce the same results as Majumdar et al. and Fu et al. using FLASH as an optimizer.

RQ2: Can we improve the runtime of classifying StackOver-flow posts?

To compare our runtime with previous work, we have computed the results of a 10 Fold Cross validation. Here, one should note that, we have implemented multi-core support for all of our experiment using the baselines of Fu and Majumdar and our experiment was run of a 4-Core machine with Intel Core i7 running Windows 10 OS. Our runtime here is for doing a 10-fold cross validation. Our experiment did a 10 fold cross validation of a 4 core machine doing 10 runs and the median values were used to make the Figure 4. As we can see, a 10 fold cross validation of DE on unclustered data takes about 5680 seconds while FLASH takes only about 845 seconds. On clustered data, DE takes 280 seconds while FLASH takes about 160 seconds. So, to answer our RQ 2, our approach using FLASH performs at least seven times faster on unclustered data and almost two times faster on clustered data while classifying StackOverflow posts.

RQ3: Can we produce a new Dataset and generate the same results?

For answering RQ3, we tried to create new dataset from Stack-Overflow posts tagged with Python. The process of creating the dataset was talked in details in Section 3.1. Our dataset had very few examples of Duplicate links and too much skewed. We tried to run our experiment over this dataset and found poor F Score for both DE and FLASH tuned SVMs. We also tried clustering our dataset using KMeans and little (6%) improvement was done. On average, a F score of 53% was seen on unclustered approach while a F score of 47% was seen on clustered approach for both DE and FLASH tuned SVMs. See Table 5 for F Score and Figure 6 for runtime. It should be noted that, although our F Scores were poor, our runtime was still significantly better than previous approach, at least eight times better on unclustered data and about one and a half times faster over clustered data.

	DE SVM			FLASH SVM			KMeans DE SVM			KMeans FLASH SVM		
Class	Precision	Recall	**F Score**	Precision	Recall	**F Score**	Precision	Recall	**F Score**	Precision	Recall	**F Score**
Duplicate	86	24	**37**	85	20	**33**	76	22	**34**	67	27	**39**
Direct	38	68	**48**	35	69	**46**	36	57	**44**	36	52	**43**
Indirect	64	55	**60**	65	54	**59**	59	52	**55**	55	52	**54**
Isolate	72	70	**71**	71	67	**69**	59	66	**63**	58	66	**62**
Average	65	55	**54**	65	53	**52**	58	50	**46**	53	49	**47**

Table 5: Python Dataset Result Comparison (RQ 3)

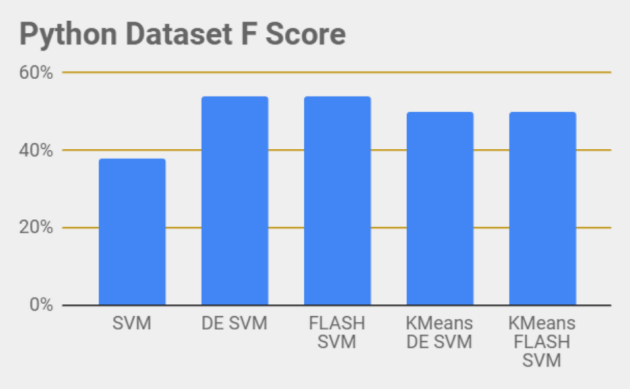
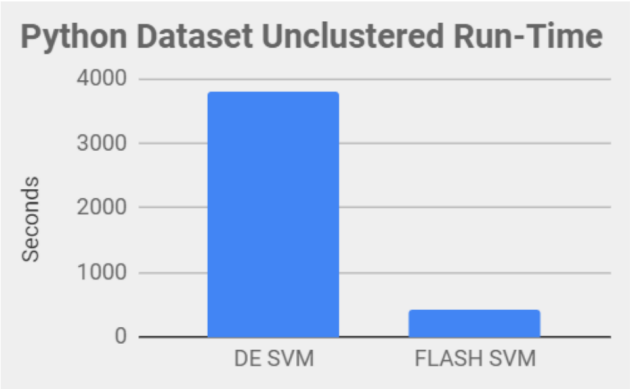


Figure 5: Python Dataset F Score Comparison (RQ3)



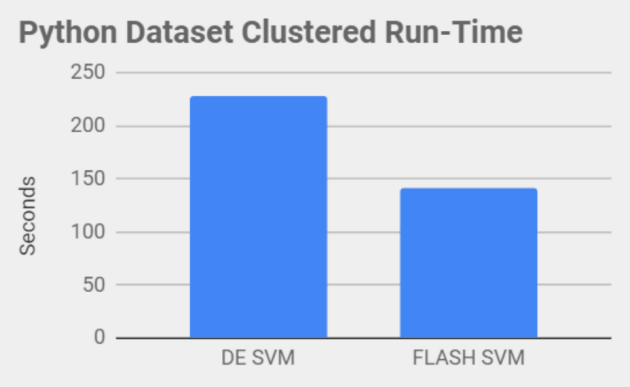


Figure 6: Python Dataset Runtime Comparision (RQ 3). Unclustered DE SVM 3794s, Unclustered FLASH SVM 440s, Clustered DE SVM 228s, Clustered FLASH SVM 141s. To understand the relative improvement, two different charts were used.

5 DISCUSSION

5.1 Baselines

Our project tergeted two baselines from the list of baselines mentioned by Sarro [14]. They were:

(1) Simple
(2) Cheap

To advocate for our improvement towards the baselines, let us first discuss the complexity of DE and FLASH. From Algorithm 1 and 2, we can see that DE takes about 20 lines of pseudo code while FLASH only takes 12 lines. Also, notice that, DE has a nested loop while FLASH is a single loop algorithm. In other words, FLASH do a one pass over the dataset while DE do double. From the point of Cyclomatic Complexity[10], FLASH has fewer branches in the control-flow than DE. If we analyze the logical complexity, DE produces new generation of data by mutation and cross-over while FLASH tries a simple prediction using CART. To summarize, FLASH is more simpler than DE and thus, our DUO approach using FLASH is simpler than the baselines.

Our RQ2 clearly shows that, FLASH is at least seven times faster than DE in non-clustered data and about twice faster than DE in clustered data. Our RQ3 also advocates for this finding. Also, see that the Algorithm 2 use a CART model that is has lower memory requirement over the frontier that DE has to hold for entire nested loop. In short, FLASH is cheaper in CPU time and in memory. So, our approach is cheaper than the baselines.

[10]Cyclomatic complexity is a software metric, used to indicate the complexity of a program. It is a quantitative measure of the number of linearly independent paths through a program's source code. It was developed by Thomas J. McCabe, Sr. in 1976. - Wiki

5.2 RQ3: Python Dataset

Our F Score for RQ3 was only about 55%. This result is not significant and we believe, the dataset that we tried to build had some issues. From our observations, there were two main reasons for this corrupted dataset.

Firstly, while trying to produce new dataset (see 3.1), we have observed that, many posts in StackOverflow "PostLink.xml" are tagged as Duplicate as well as Direct, which creates a contradiction while building Training and Testing data as we have assumed that two posts are suppose to be linked by only one type. We have also observed that, in the StackOverflow "PostLink.xml" file, often in the dataset, post X is directly linked to post Y, while post Y was not directly linked to post X. We believe, the "PostLink.xml" file has corrupt data.

Second reason behind our bad test and training set was due to the skewness of our dataset. In StackOverflow "PostLink.xml", the number of Duplicate class were only 850. So we took different portion of this class for training and testing to see our results change, but as the dataset was too skewed, it had no significant effect. Finally we had selected 500 in training set and 350 in the test set while all the other classes had 1600 in the training set and 400 in the test set. Considering our 10 fold approach in tuning the dataset, the dataset might be very skewed. As we did not do any data-balancing in our experiment, this might cause bad F scores.

5.3 Future Work

The F score and the runtime is significantly better than the Neural Network approach mentioned in Xu et al. We believe, improving upon this anymore might not show new doors in the realm of knowledge. But here are few directions that future researchers might find interesting.

As our RQ3 has corrupt data, one might try to refine the dataset and build a better dataset to test the experiment and the baselines. This way, one might not only verify our work, but also might contribute the new dataset to the community for future research.

FLASH can be a multi-objective optimization. In Nair's work[11], he showed that FLASH can use an function called BAZZA that does random vector projection to optimize for multi-objective by using multiple CARTs, one for each goal. One can optimize data miner, not only based on the F Score, but also using different matrices like IFA[11] and optimize for all of them at once.

Another possible direction of this work is to use Transfer Learning[12] by finding optimum hyperparameters from one cluster and use it in building the initial search-space of the next cluster for hyperparameter tuning.

5.4 Other

Reader might look into the results from the previous studies and see that, there is a slight improvement in in our baseline F Scores (about 2-4%). We have assumed that, this slight improvement might be due to our implementation differences or using different version of the Python libraries and assumed to be random. There is also a significant improvement in runtime of all the baselines and that is due to the machine that we were using (Intel Core-i7, 8th Gen, 4 Core) and also, converting the experiment into multi-cores. But, it should be noted that, the relative comparisons are the same.

[11]Initial False Alarm or IFA is the measurement of the inital cost of finding a terget class. Assuming that in a dataset of M size, it took k data to find the first target class. Then, all these k-1 data are false alarms and k is called the IFA measure.

[12]Transfer learning is a research problem in machine learning that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem.

6 CONCLUSION

Our project is a combination of Optimizer and Data Miner as a DUO approach. We tried tuning SVM for Text Mining using an Optimizer called FLASH over clusterd and unclustered data. We have also tried to make a new dataset using Word Embedding and traditional text preprocessing. This work shows our understanding of the course and to demonstrate that, we successfully implemented the baselines using DE and SVM over a standard Java StackOverflow dataset by Xu et al. Our baseline holds the findings of the previous studies. Then we have implemented FLASH as a hyperparameter tuner to improve Simplicity of the previous work and introduced a Cheap method overall. After comparing our results, we did static analysis to validate our findings to hold our claims. We have also tried building a new dataset to understand the breadth of the Text Mining and Optimizatin domain. Finally, we can say that, beside our improvements over the previous works, this project is a strong champion of the DUO approach.

REFERENCES

[1] Anton Barua, Stephen W Thomas, and Ahmed E Hassan. 2014. What are developers talking about? an analysis of topics and trends in stack overflow. *Empirical Software Engineering* 19, 3 (2014), 619–654.

[2] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, Feb (2012), 281–305.

[3] Ian Dewancker, Michael McCourt, and Scott Clark. 2015. Bayesian optimization primer. (2015).

[4] Danyel Fisher, Rob DeLine, Mary Czerwinski, and Steven Drucker. 2012. Interactions with big data analytics. *interactions* 19, 3 (2012), 50–59.

[5] Wei Fu and Tim Menzies. 2017. Easy over hard: A case study on deep learning. In *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM, 49–60.

[6] Chin-Jung Hsu, Vivek Nair, Vincent W Freeh, and Tim Menzies. 2018. Arrow: Low-Level Augmented Bayesian Optimization for Finding the Best Cloud VM. In *2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 660–670.

[7] Suvodeep Majumder, Nikhila Balaji, Katie Brey, Wei Fu, and Tim Menzies. 2018. 500+ times faster than deep learning (a case study exploring faster methods for text mining stackoverflow). *arXiv preprint arXiv:1802.05319* (2018).

[8] Tim Menzies, Andrew Butcher, David Cok, Andrian Marcus, Lucas Layman, Forrest Shull, Burak Turhan, and Thomas Zimmermann. 2012. Local vs. Global Lessons for Defect Prediction and Effort Estimation,âĂİ IEEE Trans. Software Eng., preprint, published online Dec. 2012; http://goo. gl/k6qno. TIM MENZIES is a full professor in computer science at the Lane. In *Department of Computer Science and Electrical Engineering, West Virginia University. Contact.* Citeseer.

[9] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.

[10] Vivek Nair, Zhe Yu, and Tim Menzies. 2017. Flash: A faster optimizer for sbse tasks. *arXiv preprint arXiv:1705.05018* (2017).

[11] Vivek Nair, Zhe Yu, Tim Menzies, Norbert Siegmund, and Sven Apel. 2018. Finding faster configurations using flash. *arXiv preprint arXiv:1801.02175* (2018).

[12] Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. ELRA, Valletta, Malta, 45–50. http://is.muni.cz/publication/884893/en.

[13] Leonard Richardson. 2013. Beautiful soup. *Crummy: The Site* (2013).

[14] Federica Sarro and Alessio Petrozziello. 2018. Linear Programming as a Baseline for Software Effort Estimation. *ACM Transactions on Software Engineering and Methodology (TOSEM)* 27, 3 (2018), 12.

[15] Rainer Storn and Kenneth Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.

[16] Robert Tibshirani, Guenther Walther, and Trevor Hastie. 2001. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63, 2 (2001), 411–423.

[17] Tianpei Xia, Rahul Krishna, Jianfeng Chen, George Mathew, Xipeng Shen, and Tim Menzies. 2018. Hyperparameter Optimization for Effort Estimation. *arXiv preprint arXiv:1805.00336* (2018).

[18] Bowen Xu, Deheng Ye, Zhenchang Xing, Xin Xia, Guibin Chen, and Shanping Li. 2016. Predicting semantically linkable knowledge in developer online forums via

convolutional neural network. In *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 51–62.